跟龙哥学真AI

rerank技术原理和实践

Sentence Transformers

Sentence Transformers开源项目：https://github.com/UKPLab/sentence-transformers

Sentence Transformers文档：https://www.sbert.net/

论文：https://arxiv.org/pdf/1908.10084

例子

安装

```
pip install -U sentence-transformers
```

```
from sentence_transformers import SentenceTransformer
sentences = ["Hello World", "Hallo Welt"]

model = SentenceTransformer('sentence-transformers/paraphrase-MiniLM-L6-v2')
embeddings = model.encode(sentences)
print(embeddings)
```

rerank模型原理

ColBERT论文：https://arxiv.org/pdf/2004.12832

ColBERT v2论文：https://arxiv.org/pdf/2112.01488

安装

```
pip install -U sentence-transformers
```

cross-encoder例子

```
from sentence_transformers import CrossEncoder
model = CrossEncoder('cross-encoder/stsb-TinyBERT-L-4')
scores = model.predict([("The weather today is beautiful", "It's raining!"),
                        ("The weather today is beautiful", "Today is a sunny
day")])


print(scores)

'''
array([0.46552283, 0.6350213 ], dtype=float32)
'''
```

bi-encorder例子

```
from sentence_transformers import SentenceTransformer

sentences=["The weather today is beautiful", "It's raining!"]
bi_encoder = SentenceTransformer('multi-qa-MiniLM-L6-cos-v1')
bi_encoder.max_seq_length = 256

corpus_embeddings = bi_encoder.encode(sentences, convert_to_tensor=True,
show_progress_bar=True)
```

llamaindex使用rerank模型

准备环境

```
#conda创建  python=3.10版本的虚环境
#conda create -n llmrag python=3.10
#激活conda创建的名字叫llmrag的虚环境
conda activate llmrag


#torch安装
conda install pytorch==2.3.1 torchvision==0.18.1 torchaudio==2.3.1 pytorch-
cuda=12.1 -c pytorch -c nvidia

#安装依赖
pip install llama_hub llama_index llama-index-readers-web trafilatura
pip install llama-index-vector-stores-chroma    #支持chroma向量数据库
pip install llama-index-embeddings-huggingface



#用openai的模型
```

```
pip install openai

#如果是本地部署ollama
pip install llama-index-llms-ollama

#通义千问线上版
pip install llama-index-llms-dashscope
```

使用网易embedding和reranker模型

使用网易有道的embedding模型bce-embedding-base_v1：https://huggingface.co/maidalun1020/bce-embedding-base_v1

```
embed_model = HuggingFaceEmbedding(model_name="maidalun1020/bce-embedding-base_v1")
```

使用网易有道的 rerank模型 bce-reranker-base_v1：https://huggingface.co/maidalun1020/bce-reranker-base_v1/tree/main

使用 SentenceTransformerRerank来加载rerank模型

```
from llama_index.core.postprocessor import  SentenceTransformerRerank
rerank = SentenceTransformerRerank( model="maidalun1020/bce-reranker-base_v1", top_n=3)

query_engine = base_index.as_query_engine(similarity_top_k=5, node_postprocessors=[rerank] )
```

完整代码

```
from llama_index.readers.web import TrafilaturaWebReader
from llama_index.core.node_parser import SimpleNodeParser
from llama_index.core.schema import IndexNode




from llama_index.embeddings.huggingface import HuggingFaceEmbedding
from llama_index.core import VectorStoreIndex, StorageContext
#from llama_index.llms.openai import OpenAI
#from llama_index.embeddings.openai import OpenAIEmbedding
from llama_index.llms.ollama import Ollama

from llama_index.core.node_parser import SentenceSplitter
from llama_index.core import Settings
import chromadb

from llama_index.core.postprocessor import  SentenceTransformerRerank
from llama_index.vector_stores.chroma import ChromaVectorStore
```

```
def prepare_data():
    url="https://baike.baidu.com/item/AIGC?fromModule=lemma_search-box"
    docs = TrafilaturaWebReader().load_data([url])
    return docs



#embed保存知识到向量数据库

def embedding_data(docs):
    #向量数据库客户端
    chroma_client = chromadb.EphemeralClient()
    chroma_collection = chroma_client.create_collection("quickstart")

    #向量数据库，指定了存储位置
    vector_store =
ChromaVectorStore(chroma_collection=chroma_collection,persist_dir="./chroma_langc
hain_db")
    storage_context = StorageContext.from_defaults(vector_store=vector_store)

    #创建文档切割器
    node_parser = SimpleNodeParser.from_defaults(chunk_size=500,chunk_overlap=50)

    #创建BAAI的embedding
    embed_model = HuggingFaceEmbedding(model_name="maidalun1020/bce-embedding-
base_v1")

    #rerank模型
    rerank = SentenceTransformerRerank( model="maidalun1020/bce-reranker-
base_v1", top_n=3)

    #创建index
    base_index = VectorStoreIndex.from_documents(documents=docs,transformations=
[node_parser],storage_context=storage_context, embed_model=embed_model)

    return base_index,embed_model,rerank



def get_llm():

    #创建OpenAI的llm
    #llm = OpenAI(model="gpt-3.5-turbo")

    #通义千问
    '''
    from llama_index.llms.dashscope import DashScope, DashScopeGenerationModels
    llm = DashScope(model_name=DashScopeGenerationModels.QWEN_MAX)
    '''

    #ollama本地模型
    llm = Ollama(model="qwen2:7b-instruct-q4_0", request_timeout=120.0)

    #创建谷歌gemini的llm
```

```
    # llm = Gemini()

    return llm

def retrieve_data(question):
    #创建检索器
    base_retriever = base_index.as_retriever(similarity_top_k=2)

    #检索相关文档
    retrievals = base_retriever.retrieve(
        question
    )

    #print(retrievals)


    #https://docs.llamaindex.ai/en/stable/examples/low_level/response_synthesis/


    from llama_index.core.response.notebook_utils import display_source_node

    for n in retrievals:
        display_source_node(n, source_length=1500)

    return retrievals


def generate_answer(question):

    query_engine = base_index.as_query_engine(similarity_top_k=5,
node_postprocessors=[rerank] )

    #大语言模型的回答
    response = query_engine.query(
       question
    )
    print(str(response))


question="艾伦•图灵的论文叫什么"
docs=prepare_data()
llm=get_llm()
base_index,embed_model,rerank=embedding_data(docs)

#通过设置来配置  llm,embedding等等
Settings.llm = llm
Settings.embed_model = embed_model
#Settings.node_parser = SentenceSplitter(chunk_size=512, chunk_overlap=20)
Settings.num_output = 512
Settings.context_window = 3000


retrieve_data(question)
generate_answer(question)

#龙 哥抖音号：龙 哥紫 貂智能
```

llamaindex使用rerank例子2

使用postprocessor的rerank

使用postprocessor来单独处理rerank流程

```
from llama_index.postprocessor.flag_embedding_reranker import
FlagEmbeddingReranker
from llama_index.schema import QueryBundle

reranker = FlagEmbeddingReranker(
    top_n = 3,
    model = "BAAI/bge-reranker-base",
)

query_bundle = QueryBundle(query_str=query)
ranked_nodes = reranker.postprocess_nodes(nodes, query_bundle = query_bundle)
```

使用LLM作为ranker

论文：https://arxiv.org/abs/2308.07107

论文 Is ChatGPT Good at Search? Investigating Large Language Models as Re-Ranking Agents：https://arxiv.org/pdf/2304.09542

列表方法详解

参考文档：https://docs.llamaindex.ai/en/stable/examples/node_postprocessor/rankGPT/

RankGPT例子代码

```
from llama_index.core.retrievers import VectorIndexRetriever
from llama_index.core import QueryBundle
from llama_index.postprocessor.rankgpt_rerank import RankGPTRerank

import pandas as pd

def get_retrieved_nodes(
    query_str, vector_top_k=10, reranker_top_n=3, with_reranker=False
):
    query_bundle = QueryBundle(query_str)
    # configure retriever
    retriever = VectorIndexRetriever(
        index=index,
```

```
        similarity_top_k=vector_top_k,
    )
    retrieved_nodes = retriever.retrieve(query_bundle)

    if with_reranker:
        # configure reranker
        reranker = RankGPTRerank(
            llm=OpenAI(
                model="gpt-3.5-turbo-16k",
                temperature=0.0,
                api_key=OPENAI_API_KEY,
            ),
            top_n=reranker_top_n,
            verbose=True,
        )
        retrieved_nodes = reranker.postprocess_nodes(
            retrieved_nodes, query_bundle
        )

    return retrieved_nodes
```

rerank模型微调

环境准备

环境我们使用和embedding微调一个环境，我们再来一遍

```
#conda创建  python=3.10版本的虚环境
conda create -n rag_embedding python=3.10
#激活conda创建的名字叫rag_embedding的虚环境
conda activate rag_embedding


#torch安装
conda install pytorch==2.3.1 torchvision==0.18.1 torchaudio==2.3.1 pytorch-
cuda=12.1 -c pytorch -c nvidia


#安装依赖
#如果用openai模型来生成微调数据集
pip install llama-index-llms-openai
pip install llama-index-embeddings-openai

#如果是本地部署ollama
pip install llama-index-llms-ollama

#安装依赖
pip install llama-index-finetuning
pip install llama-index-embeddings-huggingface
pip install -U llama-index    #要按照最新的llama-index
```

llamaindex微调文档：https://docs.llamaindex.ai/en/stable/examples/finetuning/cross_encoder_finetuning/cross_encoder_finetuning/

数据格式

参考：https://huggingface.co/datasets/sentence-transformers/msmarco-msmarco-MiniLM-L-6-v3

SentenceTransformer微调的triplet数据集格式，为jsonl文件

```
{
  "query": "what are the liberal arts?",
  "positive": 'liberal arts. 1. the academic course of instruction at a...',
  "negative": 'The New York State Education ...'
}
```

如果是4个负样本

```
{
  "query": "what are the liberal arts?",
  "positive": 'liberal arts. 1. the academic course of instruction at a...',
  "negative_1": 'The New York State Education ...',
  "negative_2": 'aaaaa...',
  "negative_3": 'bbbbb ...',
  "negative_4": 'ccccc ...',
}
```

上面是为了格式好看，真实的jsonl是一行一个json对象

```
{"query":"a","positive":"b","negative":"c"}
```

模型微调

https://sbert.net/examples/training/ms_marco/cross_encoder_README.htm

模型微调脚本也比较简单，Python代码如下：

```
import os
import logging
import pandas as pd

from torch.utils.data import DataLoader

from sentence_transformers import InputExample, LoggingHandler
from sentence_transformers.cross_encoder import CrossEncoder
from sentence_transformers.cross_encoder.evaluation import CERerankingEvaluator


DATA_DIR="G:/ai课程/rag/example/finetuning/data"
```

```
# logger
logging.basicConfig(
    format="%(asctime)s - %(message)s", datefmt="%Y-%m-%d %H:%M:%S",
level=logging.INFO, handlers=[LoggingHandler()]
)

#底模路径
model_path = "F:/sotaAI/huggingface/models--BAAI--bge-small-en"
train_batch_size = 8
num_epochs = 5
model_save_path = "ft_" + os.path.basename(model_path)


#设置num_labels=1，预测0到1之间的连续分数
model = CrossEncoder(model_path, num_labels=1, max_length=512)


train_samples = []
dev_samples = []



#读取训练数据和评估数据
train_df = pd.read_json(DATA_DIR+"/rerank_train.jsonl",lines=True)
val_df = pd.read_json(DATA_DIR+"/rerank_val.jsonl",lines=True)


for i, row in train_df.iterrows():
    train_samples.append(InputExample(texts=[row["query"],
row["positive"],row["negative"]]))


for i, row in val_df.iterrows():
    dev_samples.append(InputExample(texts=[row["query"],
row["positive"],row["negative"]]))



#构建训练的dataloader
train_dataloader = DataLoader(train_samples, shuffle=True,
batch_size=train_batch_size)

#评估器
#evaluator = CERerankingEvaluator(dev_samples, name="train-eval")

warmup_steps = 100
logging.info("Warmup-steps: {}".format(warmup_steps))


#训练模型
model.fit(
    train_dataloader=train_dataloader,
    #evaluator=evaluator,
    epochs=num_epochs,
    evaluation_steps=100,
    optimizer_params={'lr': 1e-5},
```

```
    warmup_steps=warmup_steps,
    output_path=model_save_path,
    use_amp=True
)

#保存模型
model.save(model_save_path)


#龙 哥抖音号：龙 哥紫 貂智能
```

FlagEmbedding微调rerank

文档：https://github.com/FlagOpen/FlagEmbedding/tree/master/examples/reranker

安装

```
pip install -U FlagEmbedding
```

数据格式

为jsonl文件

```
{"query": str, "pos": List[str], "neg":List[str]}
{"query": str, "pos": List[str], "neg":List[str]}
{"query": str, "pos": List[str], "neg":List[str]}
```

微调

```
torchrun --nproc_per_node {number of gpus} \
-m FlagEmbedding.reranker.run \
--output_dir {path to save model} \
--model_name_or_path BAAI/bge-reranker-base \
--train_data ./toy_finetune_data.jsonl \
--learning_rate 6e-5 \
--fp16 \
--num_train_epochs 5 \
--per_device_train_batch_size {batch size; set 1 for toy data} \
--gradient_accumulation_steps 4 \
--dataloader_drop_last True \
--train_group_size 16 \
--max_len 512 \
--weight_decay 0.01 \
--logging_steps 10
```

模型合并

文档：https://github.com/FlagOpen/FlagEmbedding/tree/master/LM_Cocktail

```
from LM_Cocktail import mix_models, mix_models_with_data

# Mix fine-tuned model and base model; then save it to output_path: ./mixed_model_1
model = mix_models(
    model_names_or_paths=["BAAI/bge-large-en-v1.5", "your_fine-tuned_model"],
    model_type='encoder',
    weights=[0.5, 0.5],  # you can change the weights to get a better trade-off.
    output_path='./mixed_model_1')
```

利用autotrain来微调rerank模型

autotrain项目是 huggingface开源的 模型训练框架工具

项目地址：https://github.com/huggingface/autotrain-advanced

autotrain文档地址：https://huggingface.co/docs/autotrain/index

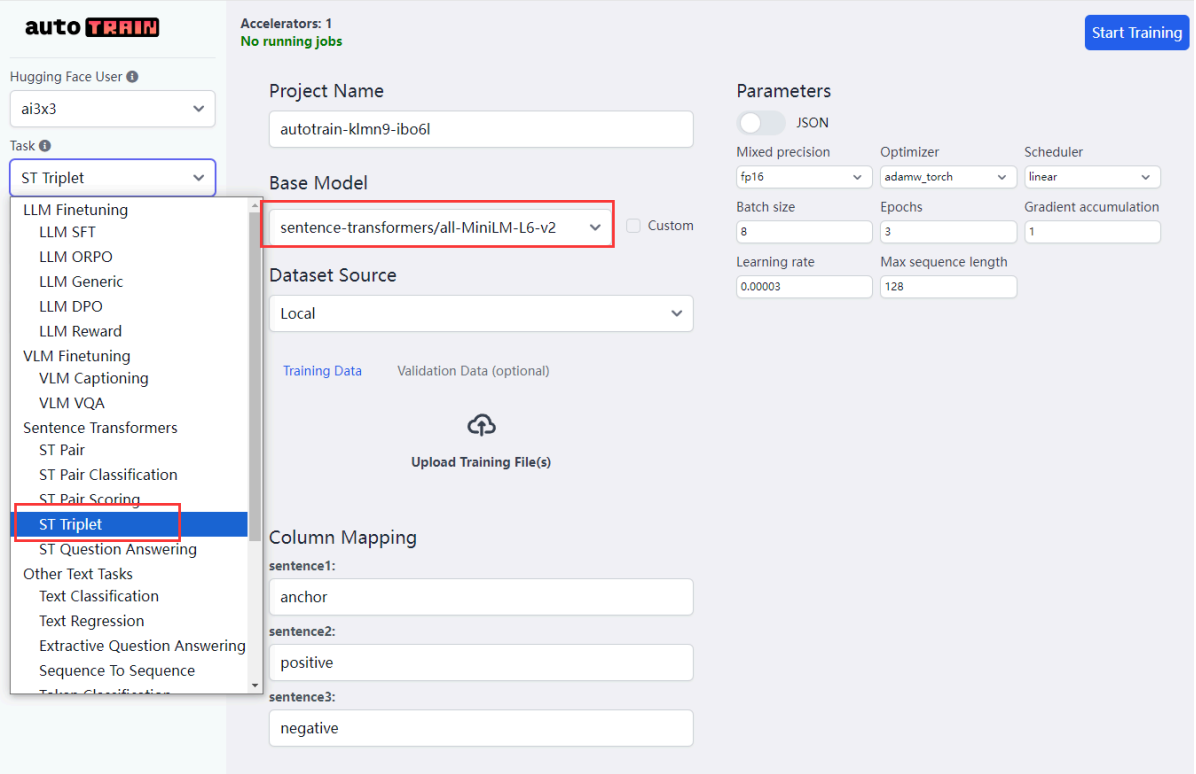
可以用下面的命令启动autorain webui界面

```
autotrain app --port 8080 --host 127.0.0.1
```

也可以用下面的命令来使用autotrain

```
autotrain --config <path_to_config_file>
```

和embedding模型中使用的基本一样，有一下几个区别：一是底模选rerank底模，二是任务要是triplet任务



rerank模型评估

mteb评估

任务列表：https://github.com/embeddings-benchmark/mteb/blob/main/docs/tasks.md

```
from mteb import MTEB
from sentence_transformers import SentenceTransformer

# Define the sentence-transformers model name
model_name = "zhengquan"

model = SentenceTransformer(model_name)
evaluation = MTEB(tasks=["MIRACLReranking"]) #指定数据集
results = evaluation.run(model, output_folder=f"results/{model_name}")
```

指定任务种类

```
evaluation = MTEB(tasks=["Reranking"])
```

c-mteb评估

c-mteb是基于mteb的

文档：https://github.com/FlagOpen/FlagEmbedding/tree/master/C_MTEB#evaluate-reranker

安装

```
pip install -U C_MTEB
```

指定c-mteb自定义的任务

```
from mteb import MTEB
from C_MTEB import *
from sentence_transformers import SentenceTransformer

# Define the sentence-transformers model name
model_name = "bert-base-uncased"

model = SentenceTransformer(model_name)
evaluation = MTEB(tasks=['T2Reranking'])   #
results = evaluation.run(model, output_folder=f"zh_results/{model_name}")
```